SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

SUSTAV AUTOMATIZIRANOG MIJEŠANJA PIĆA – DIGITALNI BARMEN

Završni rad

Ivan Živković

Osijek, 2019.

SADRŽAJ

1. UVOD

Od davnina ljudi su težili ka tome da si olakšaju život uporabom raznih pomagala i alata. To je pronalaskom električne energije i radom velikih znanstvenika kao što je Nikola Tesla prešlo na neku drugu razinu. U početku su nastali bicikli, pa prvi automobili, avioni, sve da bi se lakše kretali od točke A do točke B. Takva praksa se proširila po svim djelatnostima, prvo u tvornicama gdje imamo pokretne trake, robotske ruke, gdje je sve automatizirano kako bi se čovjeku olakšao život, a ujedno i ubrzao proces proizvodnje.

Sustavi automatiziranog miješanja pića u svojim različitim izvedbama kreću s automatizacijom proizvodnih procesa, sve točionice pića su automatizirane zbog puno veće preciznosti, brzine i učinkovitosti u odnosu na čovjeka. Tako sada u dobu kada se sve automatizira u svim područjima se gleda da se čovjek zamijeni s robotom, tako se i u ovome završnom radu zamjenjuje barmen u kafiću s njegovom digitalnom verzijom. Osoba sama sebi na web stranici kojoj pristupa preko pametnog mobitela naručuje željeno piće koje joj stroj sam priprema.

1.1 Zadatak i struktura rada

U ovom radu se radi sustav automatiziranog miješanja pića pomoću Raspberry Pi Zero W mikroračunala za upravljanje radom dva koračna motora koji će preko dozatora pića točiti zadano piće. Za web server i sustav naručivanja pića također je korišten Raspberry na kome je instaliran linux bez GUI-a.

2. POSTAV I STRUKTURA DIGITALNOG BARMENA

Automatizirani digitalni sustavi sve više zamjenjuju ljudski rad u svim djelatnostima, pa tako i u ugostiteljstvu. Primarna zadaća automatiziranih sustava nije samo da zamjene čovjekov rad već da uz to poboljša iskoristivost sustava i materijala koji se koristi za dobivanje željenog produkta. Digitalni barmen zamjenjuje jednu od uloga čovjeka u ugostiteljstvu, točnije konobara. Može biti postavljen na mnoge lokacije i korišten za različite prigode. Uloga digitalnog barmena nije samo da zamjeni čovjeka u ulozi konobara, već da posjetiteljima olakša naručivanje željenog pića uz što je moguće i praćenje potrošnje pića i rasterećenosti konobara.

2.1 Ideja i princip rada

Ideja ovog automatiziranog sustava je da njegovim korisnicima olakša naručivanje pića putem pametnog mobilnog telefona čime će smanjiti obveze konobara i ubrzati proces naručivanja pića. Naručivanje pića putem pametnog mobilnog telefona obavljalo bi se tako da se na web stranici izrađenoj posebno za ovu namjenu, koja sadrži izbor pića, odabralo željeno piće. Cilj je da stroj ne bude prevelik iz razloga lakše prenosivosti i same prilagodljivosti. Stoga je plan da se dozatori postave u kružnu formaciju radi uštede prostora i olakšanja samog pozicioniranja pića, tako da jedan elektromotor može po potrebi okretati dozatore. Pozicija dozatora iznad mjesta točenja određivala bi se preko senzora, a drugi elektromotor bi stiskao dozator kako bi se natočilo piće. Upravljanje senzorima i elektromotorima obavljalo bi mikroračunalo koje ima mogućnost bežičnog spajanja na mobilni telefon preko wi-fi modula i pokretanja web stranice.

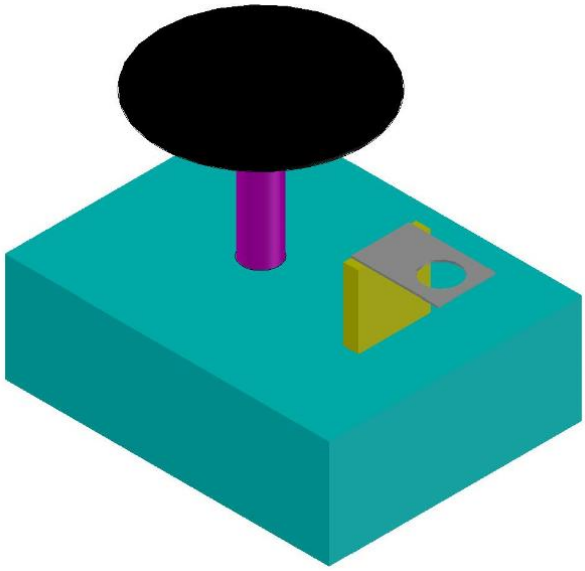
2.2 Upravljački sustav i sučelje čovjek-stroj

U složenim digitalnim sustavima poput robota, upravljački sustav je glavni i najsloženiji dio sustava. Što su zahtjevi digitalnog sustava veći, to je upravljački sustav složeniji. Upravljački sustav je računalo koje ima zadaću da primi signale i na osnovi tih primljenih signala izvrši određenu radnju.

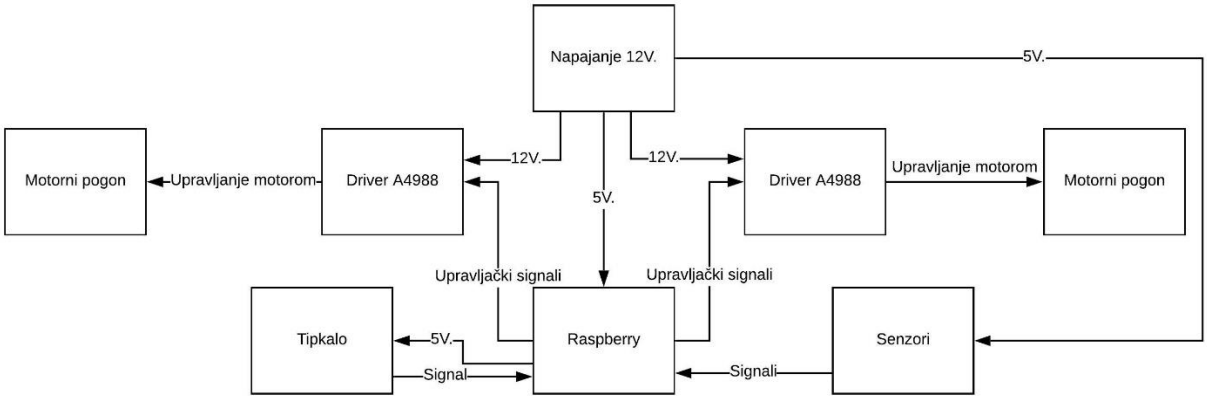
Sučelje čovjek-stroj (eng. „Human Machine Interface, HMI) je sastavni dio digitalnih sustava koji imaju mogućnost interakcije između čovjeka i stroja, odnosno digitalnog sustava. Sučelje se sastoji od sklopovlja i programske podrške koja omogućuje pretvaranje signala koji su dani od strane čovjeka na ulazu, u digitalni signal koji je namijenjen upravljačkom sustavu. Sučelje također

korisniku šalje povratni signal u obliku informacije ili potrebnog rezultata. Ovakva sučelja omogućuju uključivanje čovjeka u automatizirane digitalne sustave.

Digitalni barmen je upravljan od strane čovjeka naredbom preko pametnog mobilnog telefona koji u ovom sustavu predstavlja sučelje čovjek-stroj.



Slika 2. Skica idejnog rješenja



Slika 2.1. Strukturna shema

3. REALIZACIJA SUSTAVA

U ovom poglavlju objasnit će se tehnička realizacija sustava digitalnog barmena. Spomenut će se i objasniti sve komponente koje su bile potrebne za izradu ovog digitalnog sustava te će se dat uvid u sheme sustava.

3.1. Korištene komponente i alati

U ovom poglavlju će se govoriti o komponentama korištenim za realizaciju ovog digitalnog sustava te će se navesti njihove osnovne karakteristike.

Komponente koje su korištene za realizaciju:

- Raspberry pi zero w
- Nema 17 koračni motor
- Pololu A4988 „drive“
- Napajanje za računalo
- Tipkalo
- A3144 magnetni senzor

3.1.1 Raspberry Pi Zero W

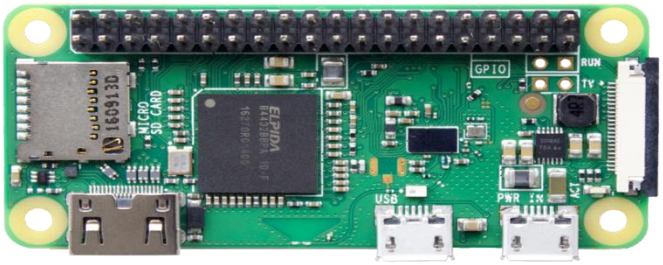
Raspberry Pi je serija mikrokontrolera razvijenih u Velikoj Britaniji od strane Raspberry Pi fondacije za promicanje podučavanja osnovnih računalnih znanosti u školama i zemljama u razvoju. Prvotni Raspberry Pi mikrokontroler je postao daleko popularniji od očekivanja zato što se prodavao na tržištu koje nije bilo ciljano te se odnosilo na uporabe poput robotike. Raspberry Pi je stekao status najprodavanijim britanskim računalom te trećim najprodavanijim računalom opće namjene.

U ožujku 2018. godine brojka prodanih mikrokontrolera dosegla je nevjerojatnih 19 milijuna. [2] Postoji nekoliko generacija Raspberry Pi mikrokontrolera. Svi modeli imaju Broadcom sustav na čipu (SoC) s integriranom ARM-kompatibilnom središnjom procesorskom jedinicom (CPU) i grafičkom procesorskom jedinicom na čipu (GPU). [2]

Specifikacije se razlikuju od modela do modela, pa tako brzine procesora se kreću od 700 Mhz do čak 1.4 Ghz uz mogućnost 64 bitnog četvero-jezgrenog ARM Cortex-A53 procesora a 512 KB „cache“ memorije. Veličine RAM memorije se kreću od 256 MB do 1 GB. Moguće je učitati

Secure Digital (SD) karticu na koju se najčešće pohranjuje operacijski sustav i programska memorija. Mikrokontroleri sadrže jedan do četiri USB porta, za video izlaz je moguć HDMI te su podržani kompozitni videozapisi, dok je zvuk moguć putem standardnog 3.5 mm audio izlaza.

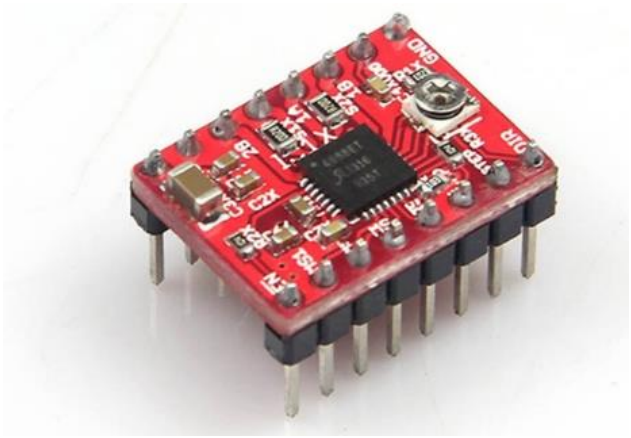
Raspberry Pi Zero W dolazi kao nadogradnja Raspberry Pi Zero koji je zbog svoje relativno male cijene i dobih performansi trenutno najpopularniji proizvod iz Raspberry Pi serije mikroračunala, on se od njega razlikuje po tome što u sebi ima ugrađen WiFi čip za bežičnu komunikaciju, kao i *bluetooth* povezivost. Napaja se sa 5V te zahtjeva dodatnu mikro SD karticu za pohranu operacijskog sustava, sastoji se od 2 mikro USB ulaza, jedan za napajanje, drugi za USB konekciju te mini HDMI konektor, Također na sebi ima i 40 raznih konektora za spajanje dodatnih elemenata i komponenti. Pokreće ga jedno jezgreni 1Ghz procesorski čip i ima 512 MB RAM memorije.



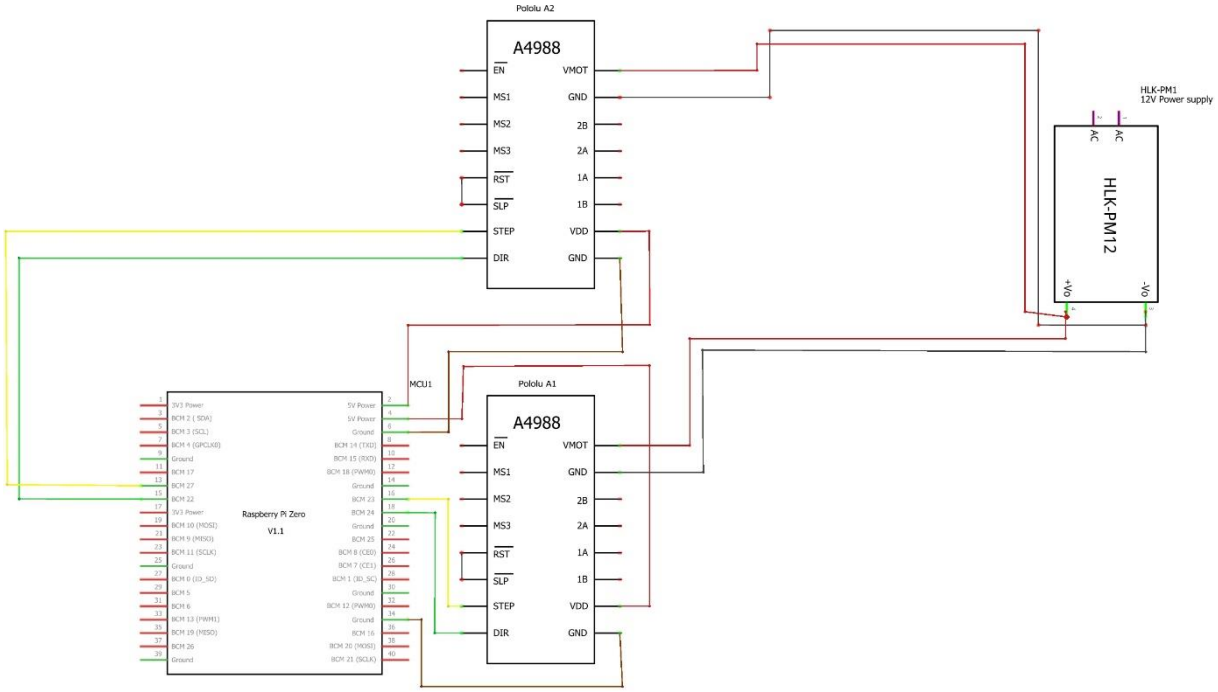
Slika 3.1. Raspberry Pi Zero W [2]

3.1.2 Pololu A4988

A4988 je ploča za pokretanje bipolarnog koračnog motora, ima dva voltažna regulatora za 3.3V i 5V da bi se izbjegla potreba za korištenjem dodatnog regulatora napona s obzirom na to da neki mikrokontroleri na izlazu daju 3.3V a neki poput Arduina 5V. Na sebi ima potenciometar za regulaciju maksimalne struje motora, ima 5 različitih koračnih rezolucija, te posjeduje regulaciju za cijeli korak, za polovinu koraka, četvrtinu, osminu i jednu šesnaestinu koraka. Može isporučiti struju do 1A po zavojnici bez dodatnog hlađenja ili do 2A s dodatnim hlađenjem. U digitalnom sustavu digitalnog barmena je potreba za snagom bila nešto veća, pa je struja podešena da 1.5 A što je zahtijevalo i dodatno hlađenje. Kako bi se omogućilo dodatno hlađenje Pololu A4988 je smješten ispred ventilatora koji je naknadno dodan i spojen na glavno napajanje digitalnog sustava.



Slika 3.1.1. Pololu A4988



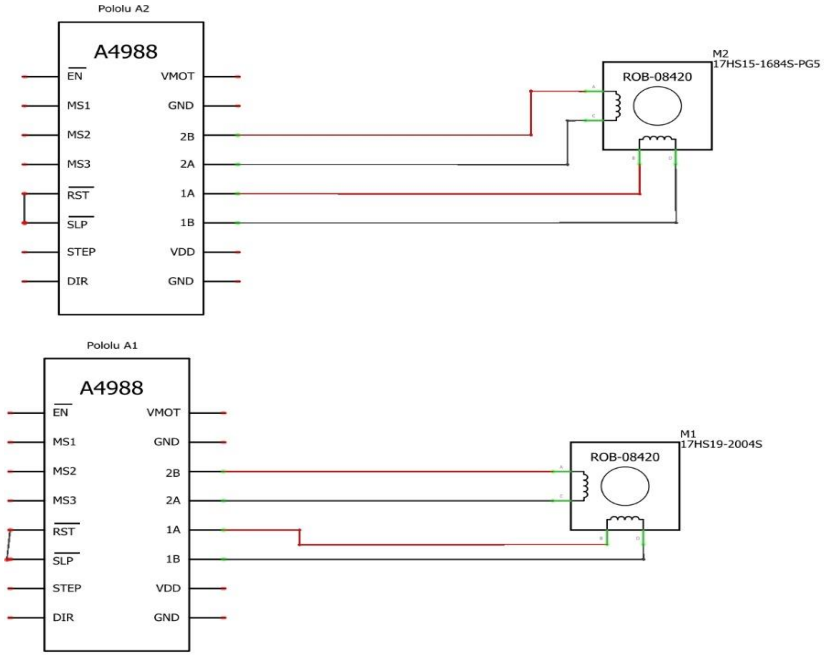
Slika 3.1.2. El. shema spajanja drivera

3.1.3 Koračni motor

Koračni motori su jako precizni, njihova preciznost proizlazi iz načina na koji su konstruirani. Njihov rotor zauzima strogo definirane i ravnomjerno raspoređene pozicije unutar punog kruga. Broj tih pozicija određuje rezoluciju koračnog motora. Koračnim motorom se upravlja u otvorenoj petlji - bez povratne veze. Odnosno ako mu se struje kroz namotaje mijenjaju prema definiranom pravilu, motor će zauzeti zadani položaj i bez naknadne provjere. Broj tih promjena određuje konačni položaj radne osovine, a brzina promjene brzinu kojom će radna

osovina dostići taj položaj. Koračni motor korišten u ovom radu ima korak od 1.8°, maksimalnu struju od 2A.

Slika 3.1.3. Nema 17 koračni motor [3]



Slika 3.1.4. El. shema spajanja motora

3.1.4 Napajanje za računalo

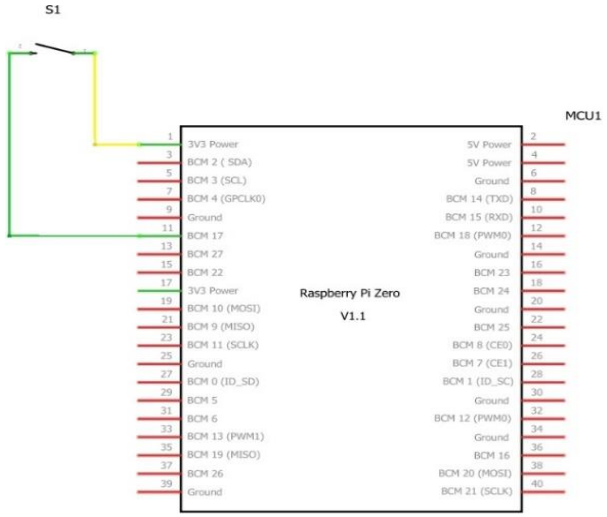
Za projekt je korišteno napajanje za računalo radi njegove stabilnosti napona i jednostavnosti. Napajanje ima izlaze od 5V i 12V i više nego dovoljnu snagu za pokretanje ovog digitalnog sustava.

Slika 3.1.5. Napajanje za računalo

3.1.5 Tipkalo

Tipkalo je korišteno za lakše upravljanje koračnim motorom za pritiskanje dozatora na tako da se nosač prvo spušta do tipkala da bi mogao uvijek za točan broj koraka pritisnuti dozator. To je napravljeno zato što prilikom transporta može doći do pomjeranja nosača koji onda ne bi obavljao svoju funkciju, a ujedno služi kao osiguranje od prebrojavanja koraka motora koje se može dogoditi ako nosač zaglavi iz nekoga razloga.

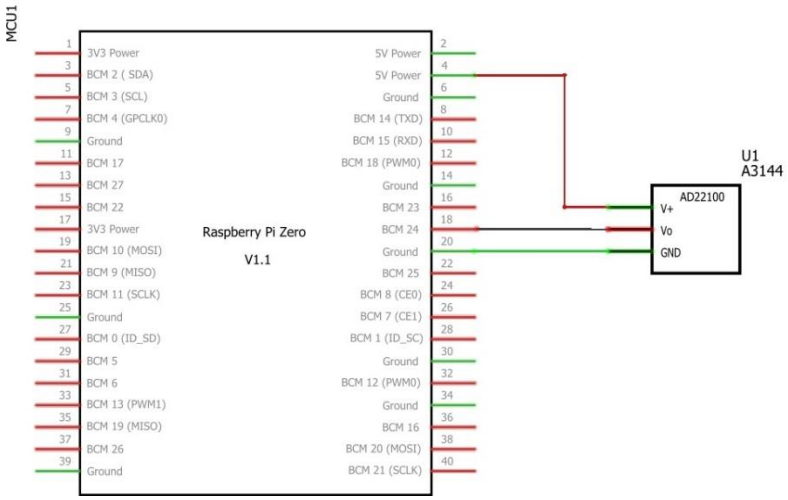
Slika 3.1.6. Tipkalo



Slika 3.1.7. El. shema tipkala

3.1.6 A3144 magnetni sensor

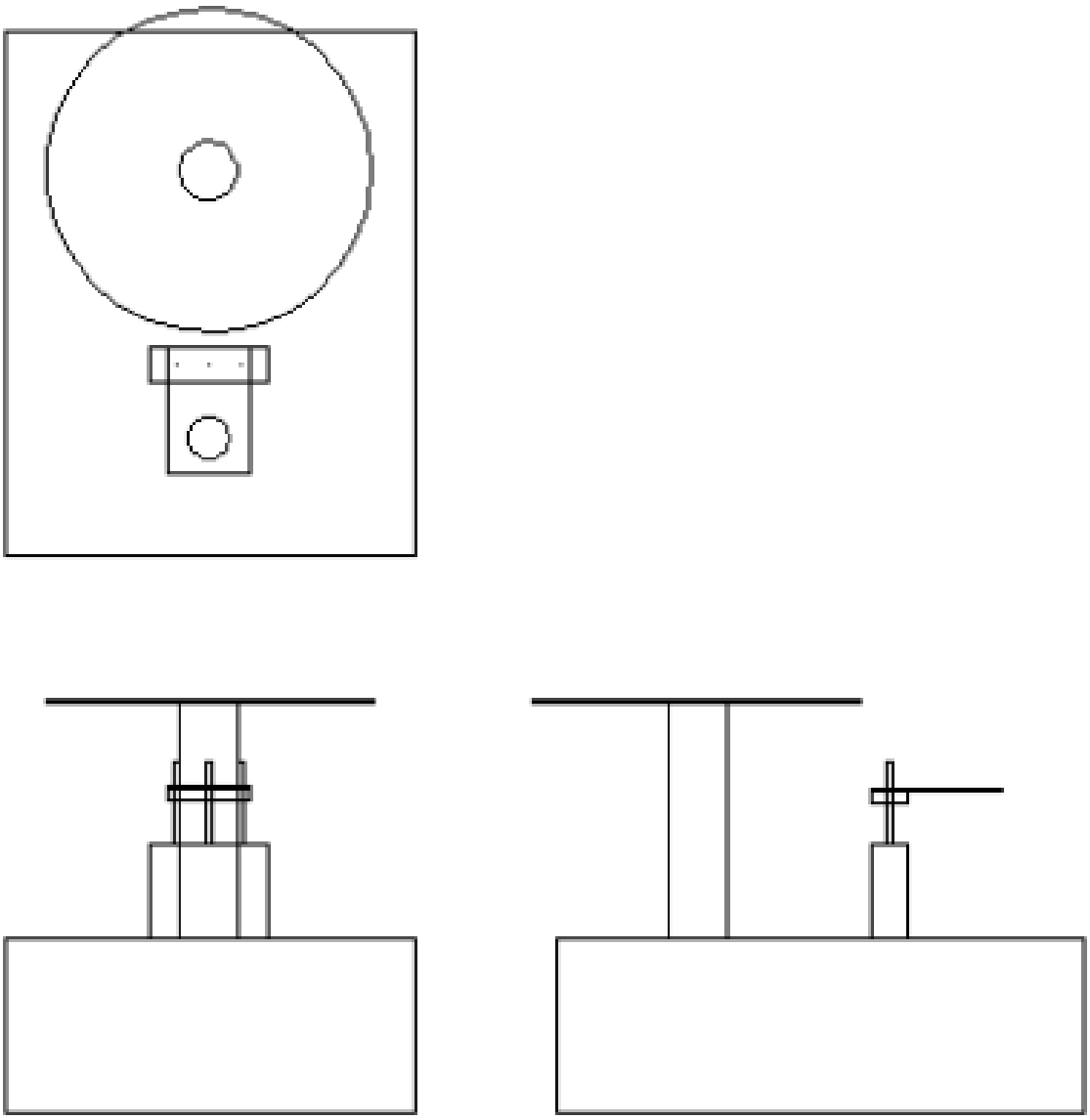
U sustavu je primijenjen magnetni senzor koji radi na principu Hallovog efekta. Ovakav magnetni senzor radi na principu da prilikom pojave magnetnog polja u blizini senzora, dolazi do pada napona na izlazu senzora. Na svaki dozator je postavljen senzor koji detektira magnetno polje magneta postavljenog na mjestu za točenje pića.



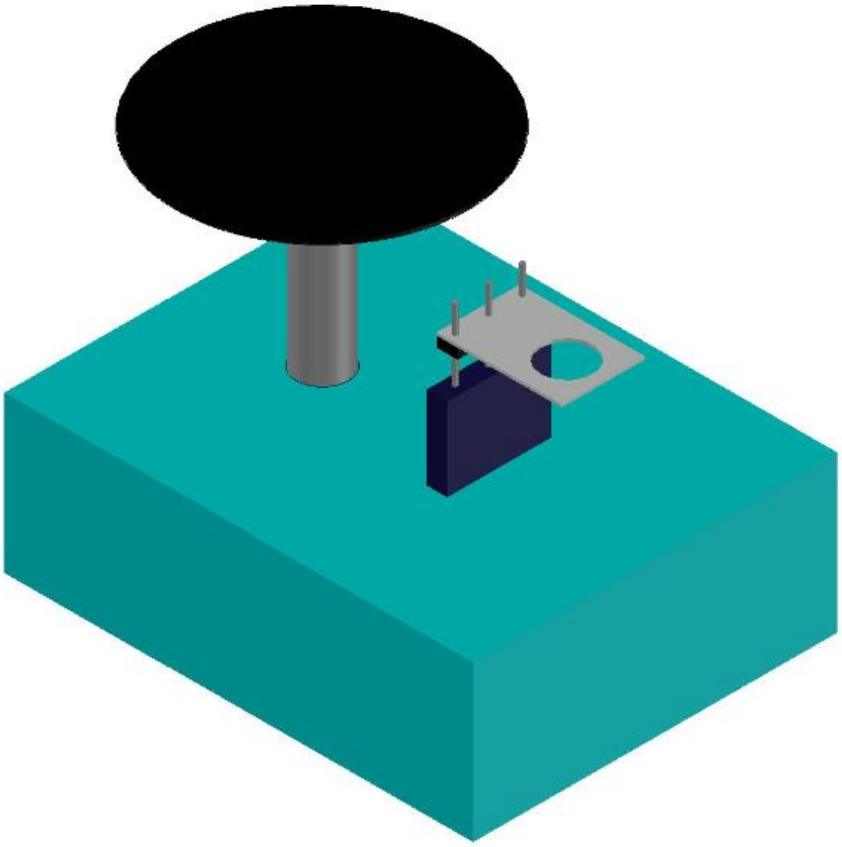
Slika 3.1.8. El. shema magnetnog senzora

3.2. Izgradnja sustava

Sustav se sastoji od metalne konstrukcije izrađene od takozvanih L – profila koja ujedno služi i kao kućište za elektroničke komponente i kablove. Plan za konstrukciju i 3D nacrt je napravljen u programskom alatu AutoCAD,, kako bi se dobila ideja kako će stoj izgledati i da ga je lakše izgraditi.



Slika 3.2. Tlocrt, nacrt, bokocrt



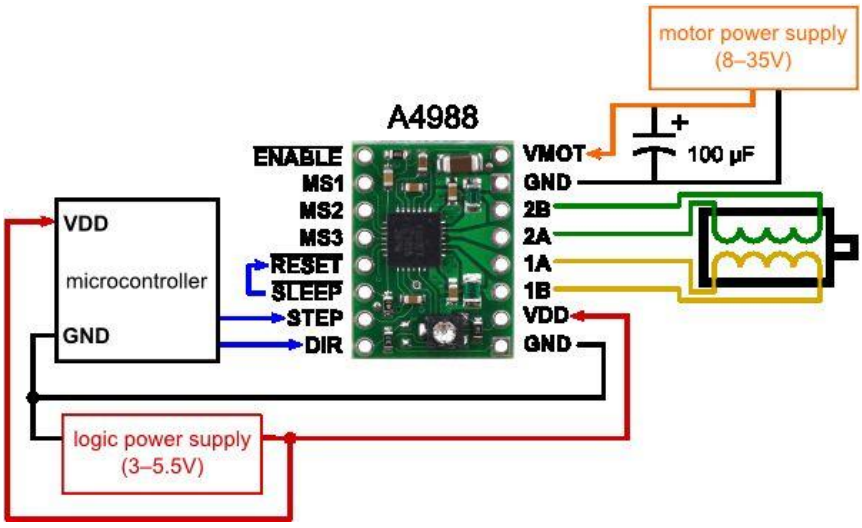
Slika 3.2.1. 3D prikaz konstrukcije

Slika 3.2.2. Izgradnja konstrukcije

Konstrukcija sadrži okrugli pokretni dio izrađen od aluminijske ploče na koji je montirano 6 dozatora za piće i 6 magnetnih senzora koji se preko posebno dizajniranog zupčanika koračnog motora vrte u krug, te se zaustavlja kada traženo piće dođe iznad čaše odnosno magneta na mjestu za točenje. Dozatori za piće su kapaciteta 30 ml te se time postiže precizno i ujednačeno točenje tijekom svake narudžbe. Pritiskanje dozatora se postiže tako da koračni motor pokreće navoj koji svojim okretanjem podiže nosač koji je napravljen od akrilnog stakla. Pokretanje i upravljanje koračnih motora obavlja se pomoću mikrokontrolera Raspberry Pi koji preko posebnog *driver-a* upravlja radom motora. „Driveri“ su smješteni ispred ventilatora koji služi za hlađenje.

Sklopovlje digitalnog barmena sadrži Raspberry Pi Zero W mikroračunalo koje omogućuje komunikaciju s web stranicom. Nakon izvršene narudžbe Raspberry Pi pokreće programsku funkciju za to piće. To se izvodi tako što jedan koračni motor okreće ploču na kojoj su postavljeni 6 dozatora te se zaustavlja kada zadano piće dođe iznad čaše, odnosno kada magnetni senzor koji je postavljen na dozator osjeti magnetno polje magneta postavljenog na mjestu za točenje, tada drugi koračni motor pritišće dozator te iz njega teče piće u čašu u određenoj mjeri.

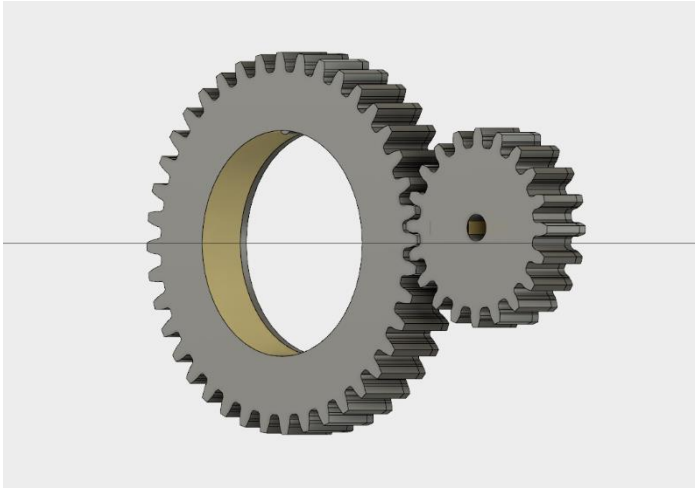
Digitalni sklopovi su ključne komponente u automatiziranim sustavima, tako i kod digitalnog barmena postoje ključne komponente koje upravljaju kretnjama sustava. Sustav se sastoji od mikroračunala Raspberry Pi Zero W koji se spaja s motorima preko *driver-a* za koračni motor Pololu A4988 na način prikazan na slici 3.22.



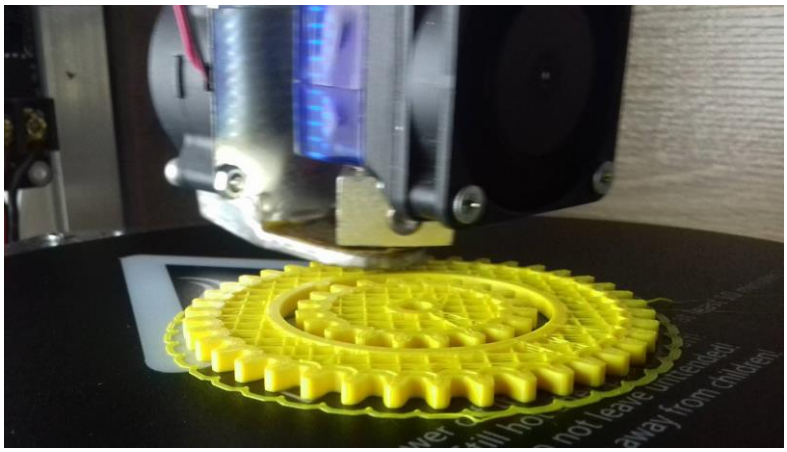
Slika 3.2.3. Spajanje mikrokontrolera s motorom [4]

Izgled cijelog sustava je prikazan na shemama u prilogu.

Realizacija okretanja ploče na koju su smješteni dozatori se postigla tako da su se izradila dva zupčanika u Autodesk Fusion 360 programskom alatu te su isprintani na 3D pisaču.



Slika 3.2.4. 3D model zupčanika



Slika 3.2.5. Proces izrade zupčanika

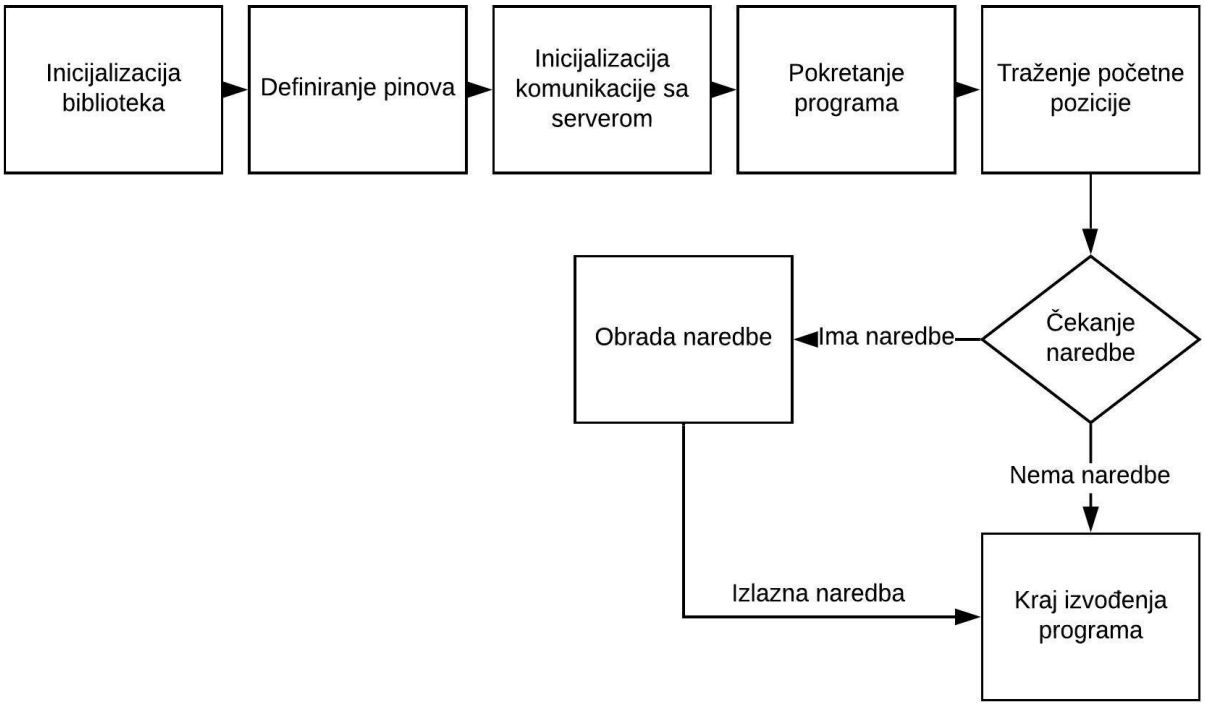
Dozatori su smješteni na ploči koja se okreće i montirani su na nosače za dozatore, koji su uz to i nosači za boce određenog pića. Boca se postavlja tako da je dno boce okrenuto prema gore, te se pomoću nosača učvrsti kako ne bi došlo do ispadanja boce. Gumeni dio na nosaču sprječava klizanje i oštećenje boce, a posebni vijak omogućuje učvršćivanje boce na nosaču.

Slika 3.2.6. Nosač za dozator i bocu

Na nosače su montirani dozatori pića tvrtke Beaumont koji ispušta dozu od 30ml. Dozatori se montiraju na donji dio nosača tako da gornji dio dozatora ulazi u bocu i time se omogućuje curenje pića u dozator. Konstrukcija dozatora je osmišljena tako da je dio koji ulazi u bocu napravljen od gume kako bi se spriječilo neželjeno curenje van boce. Tekućina iz dozatora se ispušta tako da se donji dio dozatora pritisne prema gore te se omogući istjecanje tekućine u točno određenoj količini, uz što se zatvara dotok tekućine iz boce u dozator kako bi se spriječilo neprestano curenje tekućine i mogućnost kontroliranja količine točenja pića.

Slika 3.2.7. Dozator za piće

3.3. Upravljački sustav (algoritam upravljanja)

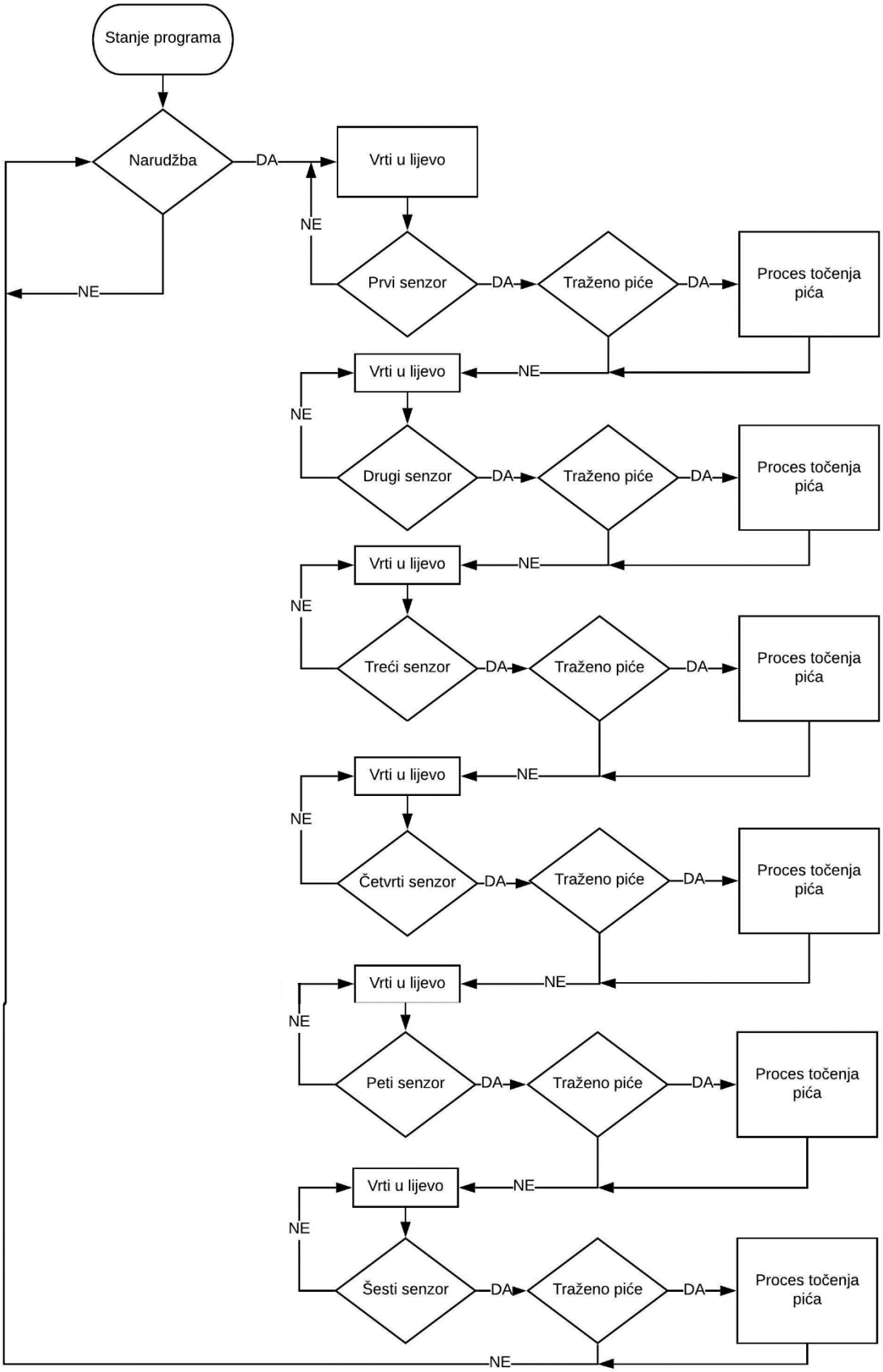


Slika 3.3. Dijagram toka sustava

U prvom dijelu programa vrši se učitavanje biblioteka potrebnih za rad sustava. Definiraju se pinovi na koje su spojeni *driveri* za koračne motore, definira se broj koraka u sekundi za motore te se upisuju definirani pinovi u program kako bi se moglo manipulirati s njima.

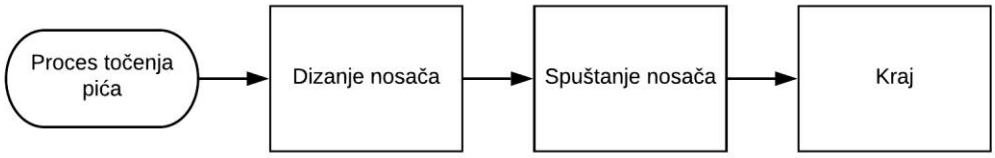
Idući dio programa je definiranje funkcija za pokretanje koračnih motora gdje im se govori u kojem smjeru se trebaju okretati i kojom brzinom tako što mijenjamo brzinu promjene polariteta na motoru, definiramo petlju koja će vrtjeti od 0 do maksimalnog broja koraka koji su definirani u DOSER_MOTOR_MOVEMENT.

U glavnom dijelu programa se definira server za komunikaciju i varijable koje će se koristiti. Također se u glavnom dijelu programa odvija primanje paketa s web aplikacije i njihova realizacija. Ako se prilikom izvršenja naredbe pošalje nova naredba za piće, ispisuje se poruka da je stroj zauzet. Ovdje se traži početna pozicija stroja tako da vrti dozatore u desno do pića broj 1. U glavnom dijelu se nalazi i logika za pokretanje motora za pritiskanje dozatora preko tipkala koje se aktivira kada nosač dođe u donju poziciju.



Slika 3.3.1. Dijagram toka obrade naredbe

Na ovome dijagramu toka prikazan je način obrade narudžbe. Kada program primi narudžbu šalje naredbu glavnome motoru za okreće dozatore u lijevu stranu do prvog senzora, te ako je to traženo piće, pokrene postupak točenja pića, ako nije motor nastavlja vrtjeti u lijevo do sljedećeg senzora. Postupak provjere se ponavlja za svaki od šest senzora i tako u krug



Slika 3.3.2. Dijagram toka procesa točenja pića

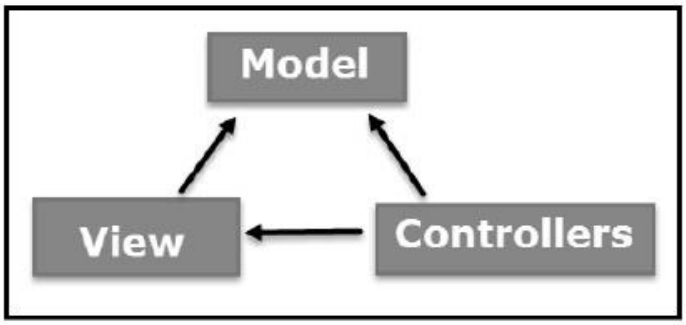
Proces točenja pića kreće kada program dobije informaciju da je traženo piće došlo na mjesto točenja odnosno senzor traženog pića došao na poziciju, tada se pokreće drugi motor koji podiže nosač do dozatora, kada dozator ispusti piće motor vrti u drugome smjeru sve dok ne dobije signal od tipkala da je došao do početnog položaja. Tako preko signala tipkala se osiguralo pouzdano točenje pića tako što će motor uvijek kretati iz iste pozicije, te ne može doći do greške čak ni nakon dugotrajnijeg korištenja.

3.4. Komunikacija i HMI sučelje

Upravljačko sučelje stroja je web aplikacija zato što se strojem želi upravljati putem svih uređaja koji su spojeni na Internet i koji mogu prikazati web stranicu, što su u ovom slučaju računalo i mobitel.

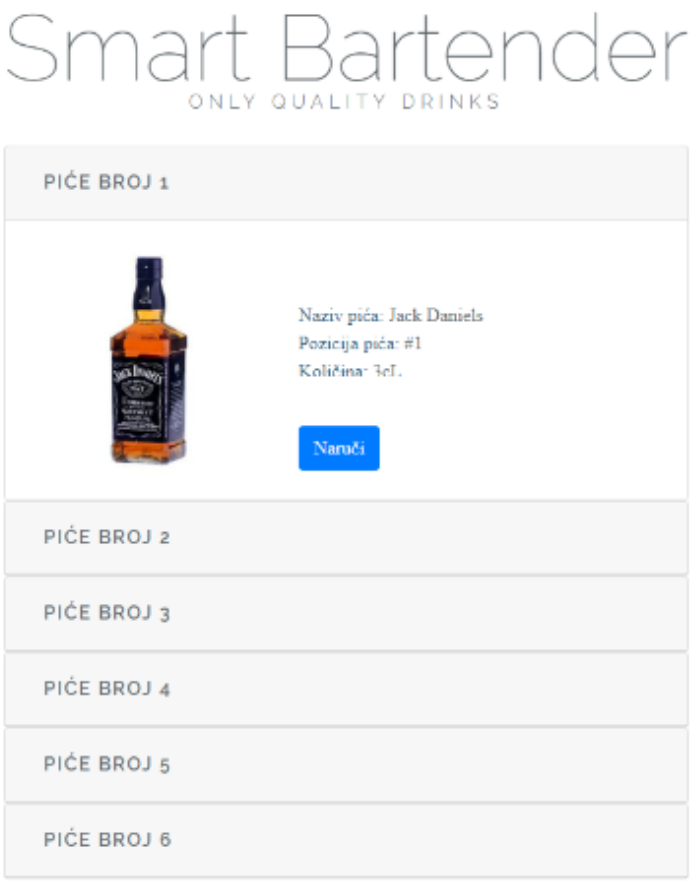
Slika 3.4. Početni ekran web aplikacije

Web aplikacije su programi koji rade na principu klijent-server odnosa gdje je klijent korisnik koji koristi web aplikaciju putem Internet preglednika. Primjeri web aplikacija su IRC, web aukcije, elektronička pošta. Web aplikacija je rađena u *model-view-controller* (MVC) arhitekturi. *Model-view-controller* (MVC) je obrazac softverske arhitekture gdje se programski kod web aplikacije dijeli na tri glavna elementa: model, *view* i *controller*.



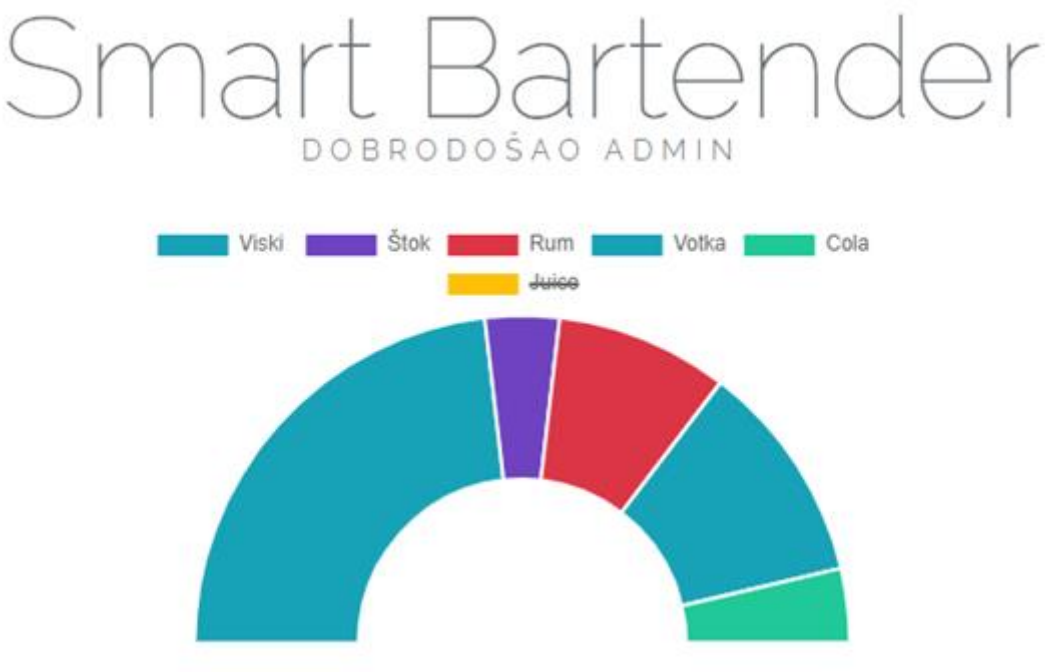
Slika 3.4.1. MVC arhitektura [7]

Model je komponenta gdje je spremljena sva logika koja ima veze s podacima. U model se mogu spremiti podaci koji se prenose između *view-a* i *controller-a* ili drugi podaci vezani uz posao. *View* je komponenta u kojoj se nalazi sva logika vezana uz grafičko sučelje aplikacije. Ovdje se nalaze HTML i CSS podaci koji prikazuju podatke dobivene iz modela na korisnikov ekran. *Controller* je komponenta koja se nalazi između modela i *view-a* koji obrađuje i manipulira podacima dobivenim iz modela, te ih šalje ka *view-u* koji tada prikazuje gotov rezultat na grafičko sučelje.



Slika 3.4.2. Naručivanje pića

Komunikacija između web servera i upravljačkog sučelja se odvija preko TCP protokola. TCP je protokol za slanje podataka putem interneta, te pruža pouzdani prijenos podataka s mjesta slanja do mjesta primitka. TCP paketi se šalju s upravljačkog sučelja, a posjeduju informacije o redoslijedu pića koje stroj mora točiti. Prije svakog slanja TCP paketa s upravljačkog sučelja ka stroju se provjerava je li stroj završio proces točenja pića, ako nije onda se ispisuje poruka na korisnikovom sučelju, a ako je onda se TCP paket može poslati ka stroju. Primljeni TCP paket se obrađuje, te se pića spremaju u posebne varijable i započinje proces točenja pića. Kada se piće naruči i uspješno pošalje ka stroju, tada se narudžba upisuje u bazu podataka koja je povezana s upravljačkim sučeljem. To administratoru omogućuje lako praćenje stanja pića putem upravljačkog sučelja. Administrator kada se jednom prijavi, njegova prijava se pamti sve do isteka sesije, nakon čega se mora ponovno prijaviti kako bi vidio stanje.



Slika 3.4.3. Izgled administratorskog panela

Kako bi se stroj imao širu primjenu dodano je da administrator kada se prijavi u sustav preko korisničkog imena i lozinke može mijenjati nazive pića, količinu kao i njihove ilustrativne slike.

Slika 3.4.4. Mijenjanje pića

4. TESTIRANJE

Kada je konstrukcija stroja sastavljena, spojeni motori i driveri te napravljena prva verzija programa, uslijedilo je testiranje i usavršavanje rada stroja. Okretanje ploče s dozatorima pravilo je problem zbog načina upravljanja motora preko njegovih koraka zato što bi se uslijed naglog kočenja pri dolasku na poziciju određenog pića ona dodatno pomaknula u stranu te dozator ne bi bio na pravom mjestu za točenje pića. Također problem je pravilo to što su se pozicije dozatora pomicale tijekom transporta i prilikom izmjene boce na dozatoru. Zbog toga je promijenjen način upravljanja glavnog motora, postavljeni su magnetni senzori na svaki dozator pa se sada uvijek zaustavlja na mjestu za točenje Koračni motor zadužen za pritiskanje dozatora trebalo je također precizno postaviti i postaviti broj koraka koji će raditi kako bi mogla stati čaša ispod njega, te kako bi efikasno stiskao dozator. Efikasnost je postignuta ugradnjom tipkala na donju poziciju nosača, motor vrti nosač prema dolje sve do signala s tipkala. Kada kontroler dobije signal s tipkala zaustavlja motor i za određen broj koraka podiže nosač. Napajanje Raspberry Pi Zero W mikrokontrolera preko glavnog napajanja za stroj je pravilo problem zbog čega se on napaja naknadno s USB mobilnim punjačem. Stroj je uspješno komunicirao s web aplikacijom.

4.1. Metode i postupci testiranja

Testiranje će se obaviti mjerenjem udaljenosti središta dozatora od središta nosača koji pritišće dozator. Zbog toga što se dozator skoro nikada ne zaustavi na istome mjestu zbog sile inercije koja dolazi do izražaja zbog povećanja mase kada je piće postavljeno na svih 6 dozatora. Također ni senzori ne reagiraju uvijek u istom trenutku te dolazi do dodatnog odstupanja. Testiranje odstupanja se obavlja kako bi bilo sigurno da će stroj uvijek nasuti piće u čašu, odnosno da će se dozator zaustaviti unutar granica otvora na nosaču. Obaviti će se i mjerenje vremena potrebnog za točenje određenog pića

4.2. Rezultati testiranja

Testiranjem odstupanja dobiveni su rezultati koji su prikazani u tablici. Dozator se uvijek zaustavljao poslije središta otvora nosača ili u rijetkim slučajevima točno na središtu. Nikada se dozator nije zaustavio prije središta otvora zato što je senzor na sredini otvora, a motor nikada ne staje prije signala sa senzora.

	Piće 1	Piće 2	Piće 3	Piće 4	Piće 5	Piće 6
Odstupanje od mjesta točenja tijekom četiri mjerenja	2 mm	1 mm	4 mm	3 mm	2 mm	0 mm
	3 mm	1 mm	3 mm	6 mm	1 mm	1 mm
	2 mm	0 mm	4 mm	5 mm	1 mm	0 mm
	2 mm	1 mm	3 mm	4 mm	1 mm	1 mm
Prosječno odstupanje	2,25 mm	0,75 m	3,5 mm	4,5 mm	1,25 mm	0,5 mm
Maksimalno odstupanje	3 mm	1 mm	4 mm	6 mm	2 mm	1 mm

Tablica 1. Odstupanje od mjesta točenja

Iz dobivenih rezultata vidljivo je da je najveće odstupanje od središta otvora bilo 6.mm te nije zadovoljilo uvjet da sipa piće u čašu. Zato je otvor nosača dodatno proširen na polumjer od 1.cm kako bi se osiguralo da piće uvijek bude ispravno natočeno.

	Piće 1	Piće 2	Piće 3	Piće 4	Piće 5	Piće 6	Kombinacija
Vrijeme[s]	7	15	20	24	28	32	40

Tablica 2. *Vrijeme točenja pića*

Mjerenje vremena točenja pića je obavljeno radi saznanja brzine točenja pojedinog pića, kao i neke od kombinacija pića.



Slika 4. Izgled gotovog stroja

5. ZAKLJUČAK

Izvršavanje ovog projekta zahtijevalo je mnogo truda i znanja. Za sam početak i izgradnju stroja bilo je potrebno savladati tehniku elektrolučnog zavarivanje L -profila, također je trebalo dizajnirati zupčanike i konstrukciju u računalnom programu za 3D ispis te njihova realizacija pomoću 3D pisača. Bilo je potrebno savladati način rada koračnih motora te uređaja (*drivera*) za njihov pogon i upravljanje. Također je prošireno znanje korištenja Linux sustava stečeno na fakultetu kao i programiranja u programskom jeziku python. Za kraj crtanje električnih shema za dokumentaciju obavljeno je u programskom alatu „Fritzing“. Znanje stečeno pri izradi ovog projekta će služiti daljnjem usavršavanju i radu u struci automatičara.

6.LITERATURA

[1] HMI interface, https://www.techopedia.com/definition/12829/human-machine-interface-hmi [20.09.2019]

[2] Raspberry Pi Zero W, https://en.wikipedia.org/wiki/Raspberry_Pi [16.08.2019]

[3] Koračni motor, https://e-radionica.com/hr/nema17-stepper-motor.html [16.08.2019]

[4] Pololu A4988 modul, https://www.pololu.com/product/1182 [16.08.2019]

[5] TCP protokol, https://en.wikipedia.org/wiki/Transmission_Control_Protocol [20.08.2019]

[6] Web aplikacija, https://en.wikipedia.org/wiki/Web_application [20.08.2019]

[7] MVC arhitektura, https://hr.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller [20.08.2019]

SAŽETAK

Naslov: Sustav automatiziranog miješanja pića-digitalni barmen

U ovome radu napravljen je stroj za automatizirano točenje pića, opisan postupak izgradnje i rada stroja. Za početak bilo je potrebno napraviti čvrstu i ne previše veliku konstrukciju, kada je to riješeno sve komponente su spojene na mikrokontroler Raspberry Pi Zero W koji služi za upravljanje koračnim motorima za točenje pića. Mikrokontroler se koristi i za upravljanje web stranicom tako što ima ugrađen wi-fi modul preko kojega se spaja na mrežu. Na kraju rada su rezultati rada gdje je stroj testiran i gdje su obavljena mjerenja pouzdanosti i brzine točenja pića. Prikazan je izgled samog stroja i objašnjeno kako je postignuta bolja učinkovitost.

Ključne riječi: Dozator pića, Raspberry Pi, Koračni motor, Magnetni senzor

ABSTRACT

Title: Automated Beverage Mixing System – Digital Barmen

In this paper, an automatic beverage dispensing machine is made, the procedure for construction and operation of the machine is described. To begin with, it was necessary to make a solid and not too large structure, when this was resolved all components were connected to a Raspberry Pi Zero W microcontroller, which is used to control stepper motors for refilling drinks. The microcontroller is also used to manage the web site by having a built-in wi-fi module through which it connects to the network. At the end of the work are the results of the operation where the machine was tested and where the reliability and speed measurements were made. The layout of the machine itself is shown and explained how better performance is achieved.

Keywords: Beverage dispenser, Raspberry Pi, Stepper motor, Magnetic sensor

ŽIVOTOPIS

Ivan Živković rođen je 31. prosinca 1996. godine u Slavonskom Brodu. Ima završenu srednju elektrotehničku školu u Orašju, Bosna i Hercegovina. Čita i piše dva strana jezika, engleski i njemački. Odradio je stručnu praksu u TEO-Belišće u Belišću.

_____

Potpis

PRILOG

Kod za upravljanje strojem:

```
from time import sleep
import OPi.GPIO as GPIO
import servers as server
# PINS
DOSER_MOTOR_DIR        = 8
DOSER_MOTOR_STEP       = 10
PUSH_BUTTON                    = 3
DRINKS_MOTOR_DIR       = 36
DRINKS_MOTOR_STEP      = 38
DRINK_SENSOR_1         = 29
DRINK_SENSOR_2         = 33
DRINK_SENSOR_3         = 40
DRINK_SENSOR_4         = 37
DRINK_SENSOR_5         = 35
DRINK_SENSOR_6         = 31


# Steps per revolution
DOSER_MOTOR_SRP = 200
                # (360/1.8)
DOSER_MOTOR_MOVEMENT = DOSER_MOTOR_SRP * 35
        # 35 turns
DRINKS_MOTOR_FULL_SRP = 1029
        # (360/0.35=1029)
```

```
DRINKS_MOTOR_SRP = 172
                # 1029/6=172
DRINKS_MOTOR_MOVEMENT = DRINKS_MOTOR_SRP * 2
# Main Loop
socket = server.SocketServer()
socket.listen()
drink_1 = -1
drink_2 = -1
current_drink = -1
last_slot = 0
is_on_drink = False
barmen_state = "finding_start_point"
is_doser_down = True
def sensorCallback1(channel):
global barmen_state, is_on_drink, current_drink
MOTOR_DRINK_PLACED_1 = True if GPIO.input(DRINK_SENSOR_1) == False else False
if MOTOR_DRINK_PLACED_1 == True:
if barmen_state == "finding_start_point":
stop_motors()
is_on_drink = True
current_drink = -1
barmen_state = "waiting_order"
elif current_drink == 1:
                stop_motors()
is_on_drink = True
```

```
current_drink = -1
        barmen_state = "dosing_drinks_up"
def sensorCallback2(channel):
global barmen_state, is_on_drink, current_drink
if barmen_state == "finding_start_point":
return
MOTOR_DRINK_PLACED_2 = True if GPIO.input(DRINK_SENSOR_2) == False else False
if MOTOR_DRINK_PLACED_2 == True and current_drink == 2:
stop_motors()
                                        is_on_drink = True
current_drink = -1
barmen_state = "dosing_drinks_up"
def sensorCallback3(channel):
global barmen_state, is_on_drink, current_drink
if barmen_state == "finding_start_point":
return
MOTOR_DRINK_PLACED_3 = True if GPIO.input(DRINK_SENSOR_3) == False else False
if MOTOR_DRINK_PLACED_3 == True and current_drink == 3:
stop_motors()
is_on_drink = True
current_drink = -1
barmen_state = "dosing_drinks_up"
def sensorCallback4(channel):
global barmen_state, is_on_drink, current_drink
if barmen_state == "finding_start_point":
```

```
return
MOTOR_DRINK_PLACED_4 = True if GPIO.input(DRINK_SENSOR_4) == False else False
if MOTOR_DRINK_PLACED_4 == True and current_drink == 4:
stop_motors()
is_on_drink = True
current_drink = -1
barmen_state = "dosing_drinks_up"
def sensorCallback5(channel):
global barmen_state, is_on_drink, current_drink
if barmen_state == "finding_start_point":
return
MOTOR_DRINK_PLACED_5 = True if GPIO.input(DRINK_SENSOR_5) == False else False
if MOTOR_DRINK_PLACED_5 == True and current_drink == 5:
stop_motors()
is_on_drink = True
current_drink = -1
barmen_state = "dosing_drinks_up"
def sensorCallback6(channel):
global barmen_state, is_on_drink, current_drink
if barmen_state == "finding_start_point":
return
MOTOR_DRINK_PLACED_6 = True if GPIO.input(DRINK_SENSOR_6) == False else False
if MOTOR_DRINK_PLACED_6 == True and current_drink == 6:
stop_motors()
```

```
is_on_drink = True

current_drink = -1

barmen_state = "dosing_drinks_up"

def pushButtonCallback(channel)

global is_doser_down

if is_doser_down == True:

return

stop_motors()

is_doser_down = True

# GPIO Settings

GPIO.setboard(4)

GPIO.setmode(GPIO.BOARD)

GPIO.setup(DRINKS_MOTOR_DIR, GPIO.OUT)

GPIO.setup(DRINKS_MOTOR_STEP, GPIO.OUT)

GPIO.setup(DOSER_MOTOR_DIR, GPIO.OUT)

GPIO.setup(DOSER_MOTOR_STEP, GPIO.OUT)

GPIO.setup(DRINK_SENSOR_1 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(DRINK_SENSOR_1, GPIO.BOTH, callback=sensorCallback1, bouncetime=200)

GPIO.setup(DRINK_SENSOR_2 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(DRINK_SENSOR_2, GPIO.BOTH, callback=sensorCallback2, bouncetime=200)

GPIO.setup(DRINK_SENSOR_3 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(DRINK_SENSOR_3, GPIO.BOTH, callback=sensorCallback3, bouncetime=200)
```

```
GPIO.setup(DRINK_SENSOR_4 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(DRINK_SENSOR_4, GPIO.BOTH, callback=sensorCallback4, bouncetime=200)

GPIO.setup(DRINK_SENSOR_5 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(DRINK_SENSOR_5, GPIO.BOTH, callback=sensorCallback5, bouncetime=200)

GPIO.setup(DRINK_SENSOR_6 , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(DRINK_SENSOR_6, GPIO.BOTH, callback=sensorCallback6, bouncetime=200)

GPIO.setup(PUSH_BUTTON , GPIO.IN, pull_up_down=GPIO.PUD_UP)

GPIO.add_event_detect(PUSH_BUTTON, GPIO.RISING, callback=pushButtonCallback)

# Functions

def doser_motor_movement(direction):

GPIO.output(DOSER_MOTOR_DIR, direction)

for x in range(DOSER_MOTOR_MOVEMENT):

GPIO.output(DOSER_MOTOR_STEP, GPIO.HIGH)

sleep(0.002)

GPIO.output(DOSER_MOTOR_STEP, GPIO.LOW)

def drinks_motor_movement(direction):

global is_on_drink

GPIO.output(DRINKS_MOTOR_DIR, direction)

for x in range(DRINKS_MOTOR_MOVEMENT):

if is_on_drink == True:

break

GPIO.output(DRINKS_MOTOR_STEP, GPIO.HIGH)
```

```
sleep(0.003)
GPIO.output(DRINKS_MOTOR_STEP, GPIO.LOW)
def drinks_looped_movement(movement, direction):
global is_on_drink, barmen_state, last_slot
last_slot = movement
for x in range(movement):
if is_on_drink == True:
break
drinks_motor_movement(direction)
sleep(0.001)
def stop_motors():
global is_on_drink
is_on_drink = True
GPIO.output(DRINKS_MOTOR_STEP, GPIO.LOW)
GPIO.output(DRINKS_MOTOR_DIR, GPIO.LOW)
GPIO.output(DOSER_MOTOR_STEP, GPIO.LOW)
print("Zapocinjem")
try:
while True:
if(socket.data != ""):
data = socket.data[0].split(":")
if drink_1 != -1 or drink_2 != -1:
socket.send_message("2015")
socket.data = ""
```

```
continue
if data[0] != "drink":
socket.data = ""
continue
if len(data) == 3:
drink_1 = int(data[1])
drink_2 = int(data[2])
elif len(data) == 2:
drink_1 = int(data[1])
socket.send_message("2810")
socket.data = ""
if barmen_state == "finding_start_point":
drinks_looped_movement(10, 1)
elif barmen_state == "waiting_order":
is_on_drink = False
current_drink = -1
if drink_1 != -1:
current_drink = drink_1
elif drink_2 != -1:
current_drink = drink_2
if current_drink == -1:
continue
drinks_looped_movement(10, 0)
```

```
elif barmen_state == "dosing_drinks_up":
is_doser_down = False
doser_motor_movement(1)
sleep(1)
barmen_state = "dosing_drinks_down"
elif barmen_state == "dosing_drinks_down":
doser_motor_movement(0)
sleep(1)
if drink_1 != -1 and drink_2 != -1 and drink_1 == drink_2:
barmen_state = "dosing_drinks_up"
drink_1 = -1
drink_2 = -1
else:
if drink_1 != -1:
drink_1 = -1
elif drink_2 != -1:
drink_2 = -1
sleep(0.1)
barmen_state = "waiting_order"
elif barmen_state == "exit":
break
except KeyboardInterrupt:
pass
print("Zavrseno!")
```

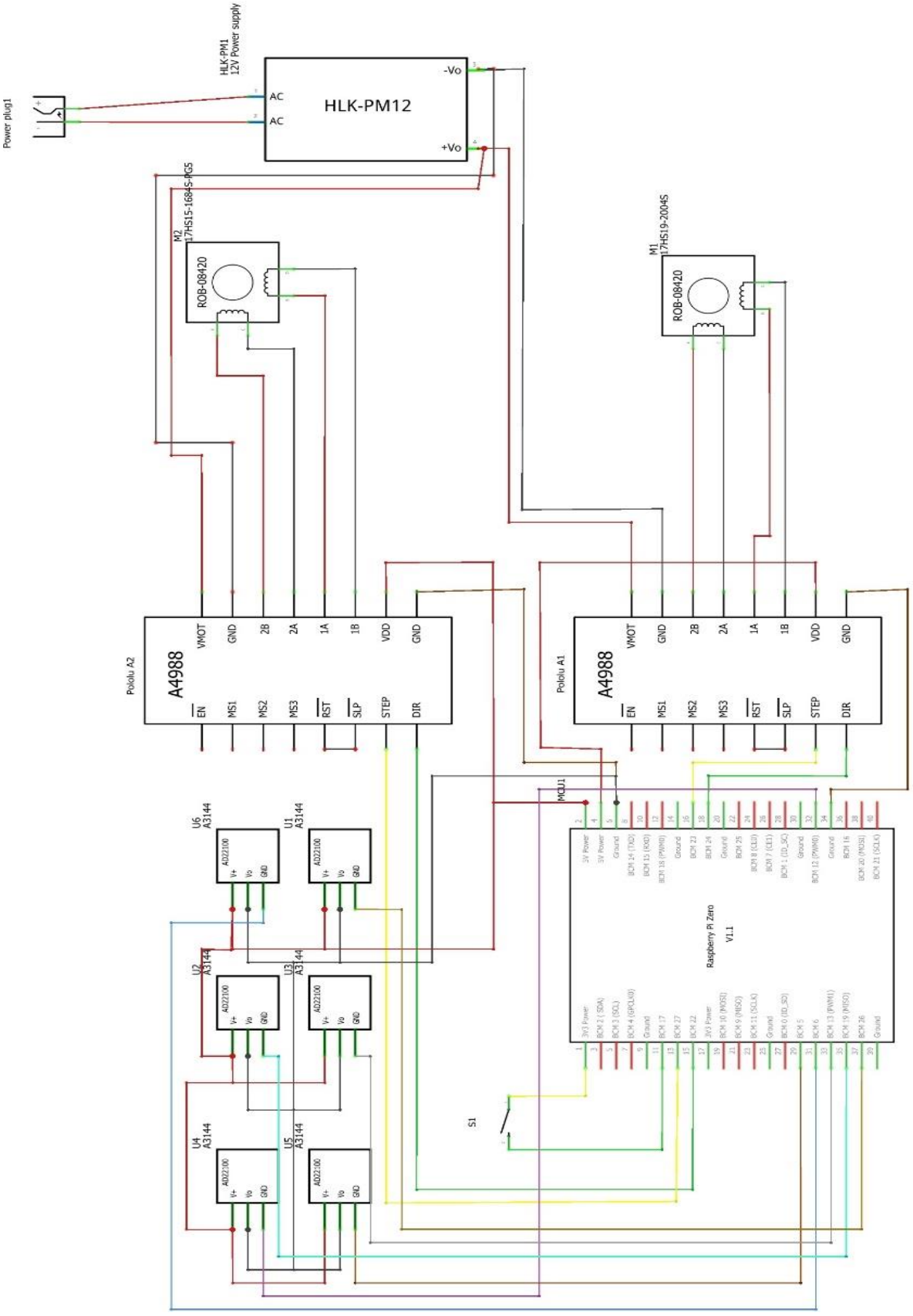
```
stop_motors()

GPIO.cleanup()

socket.close()
```

Kod za komunikaciju sa serverom:

```
import socket

from threading import Thread

import time

class SocketServer:

IP = "localhost"

PORT = 5050

BUFFER_SIZE = 10


def __init__(self):

self.sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

self.sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

self.sock.bind((self.IP, self.PORT))

self.data = ""

self.client = None

self.started = False

def send_message(self, data):

msg = data

self.client.sendall(msg.encode('utf-8'))
```

```
def recv_message(self, client):

data = client.recv(self.BUFFER_SIZE)

decoded_data = data.decode("utf-8")

splited_data = decoded_data.split("\\n")

return splited_data

def listen(self):

if self.started:

return None

self.started = True

self.sock.listen(1)

self.socketThread = Thread(target = self.listen_clients, args = ())

self.socketThread.start()

def listen_clients(self):

while self.started:

client, addr = self.sock.accept()

self.client = client

self.data = self.recv_message(client)

def close(self):

self.sock.shutdown(socket.SHUT_RDWR)

self.sock.close()

self.started = False
```

Električna shema



Montažna shema